Sparsification of CSC data

Michael Schernau

September 7, 1999

Abstract

This note describes a block diagram of a possible hardware implementation of the sparsifier for the CSC data readout.

1 Introduction

Data from the CSC consists of digitized pulse heights of groups of adjacent strips. These clusters are used to reconstruct the position of the incident track(s). The signals of the strips between the clusters are digitized as well, but suppressed by the sparsifier logic. This data reduction is necessary to transmit the meaningful data within the available bandwidth. Even at a flux of 1500 Hz/cm^2, five times higher than the expected flux, the probability for a hit cluster in one layer is only 3/8.

In addition to suppressing channels without hits, the sparsifier also suppresses signals which have started before or after the trigger. The rejection of these wrong-time pulses is essential for limiting the bandwidth of the data stream.

2 Algorithm

The algorithm for the sparsification operates on the four consecutive time samples of each channel. These samples are spaced 25 ns apart and called A, B, C and D. The peak of the signal of average drift time lies between sample B and C. The phase of the sampling should be adjustable to ensure this condition.

In order to reject the channels without hits, a threshold is applied to the two biggest samples, B and C. This threshold is adjustable to optimize selection efficiency and rejection rates.

Wrong-time pulses can be rejected by requiring a rising slope between samples A and B and a falling slope between samples C and D. This requirement results in an acceptance window that is two time bins wide. This width of 50 ns is slightly larger than the maximum drift time of 35ns, ensuring a high selection efficiency over the entire range of drift times.

In summary, with T denoting the threshold, the selection criteria are:

- $B > T$
- $C > T$
- $A < B$
- $C > D$

In the final step of the Sparsification, clusters are formed from the list of channels that satisfy the selection criteria. This is done by simply marking one or two adjacent channels on either side of the hit channel. These channels might contain charge information that is important for the reconstruction of the track position, even though the signal did not satisfy the threshold condition.

3 Block Diagram

Figure 1 shows the block diagram of the sparsification hardware. Its central part is a memory buffer which contains the digitized samples. Three major blocks access this memory: A write logic stores new samples, a check logic reads the samples and applies the selection criteria and a readout logic reads the selected samples, reformats them, and writes the output into a FIFO.

The memory is divided into two parts: one part is written to, and the other is read from. This ensures that the reading and checking block can work on a complete set of samples.

The memory has to be able to service one write request and two read requests within the time between two incoming samples. It therefore has to run at a frequency several times higher than that of the ADC.

3.1 Write Logic

The purpose of the write logic is to write the digitized samples into the buffer. The writing process is started when a new trigger is received or when a previous trigger has not yet been processed. Incoming triggers are counted in the trigger counter. It is decremented after all data pertaining to a trigger has been read out.

The readout process starts the sample counter, which counts from zero up to four times the number of channels. Two addresses are derived from this counter: the SCA address and the buffer address. The SCA address tells the SCA controller which sample to present to the ADC for digitization, while the buffer address is used to store the result in the sample buffer. The ADC value itself is buffered in a register and possibly extended to 16 bit if the buffer is 16 bit wide.

3.2 Check Logic

The check logic has its own sample counter. A buffer address is derived from it and the ADC value is read from the buffer. This value is stored in the A, B, C, or D register, depending on its sample time. Once all four samples for a channel are loaded, the comparator makes the following four comparisms: $B > T$, $C > T$, $A < B$, and $C > D$. The results are fed into an and gate. An external mask of selection criteria can mask out some or all criteria for test and calibration purposes.

The output of the and gate is fed into the set neighbors logic. This logic block sets a bit for this and the adjacent channels in the hit mask. An external input (Width) specifies the number of neighbors to set. It is expected to set two adjacent bits on each side of a channel for the precision strips and one for the transverse strips.

A channel mask makes it possible to mask out "hot" channels. These channels will be read out if they are part of a cluster, but they can not start a cluster by them-self.

The hit mask contains a flip-flop for every channel that the module services (for instance, 12 channels). Two additional flip-flops are added on either side of the array. These flip-flops can be set by the set neighbors logic if a cluster begins or ends near the boundary of the array. These four bits are exported to the adjacent sparsifiers where they indicate a continuation of the cluster. Conversely, each sparsifier has four inputs where adjacent sparsifiers can request channels to be read out as a continuation of their clusters. These inputs set the bits 0, 1, $n-2$, and $n-1$ in the $2+n+2$ bit hit mask.

3.3 Read Logic

This logic group reads the channels that are marked in the hit mask, reformats the data, and writes it into the output FIFO. Operation of this block starts after the check logic has completed its operation and reads the same sample data from the buffer.

A sample counter counts from zero to $4n$. The buffer address for the sample is generated and the ADC value for the sample is read out. At the same time, the hit mask is checked. If the corresponding bit is not checked, then no further action is taken and the next ADC value overwrites the current one. Otherwise, the ADC value is put into the current sample word. It is expected to fit two or three ADC values into one sample word. Once the sample word is completed, it is written into the output FIFO.

The new cluster logic checks if a new cluster starts. In this case it reads the channel counter and creates a cluster word indicating the channel number of the first channel of the new cluster. This cluster word is written into the output FIFO. The logic doesn't generate a cluster word if an external request for channel 0 is received because it is assumed that the requesting sparsifier has already created a cluster word.

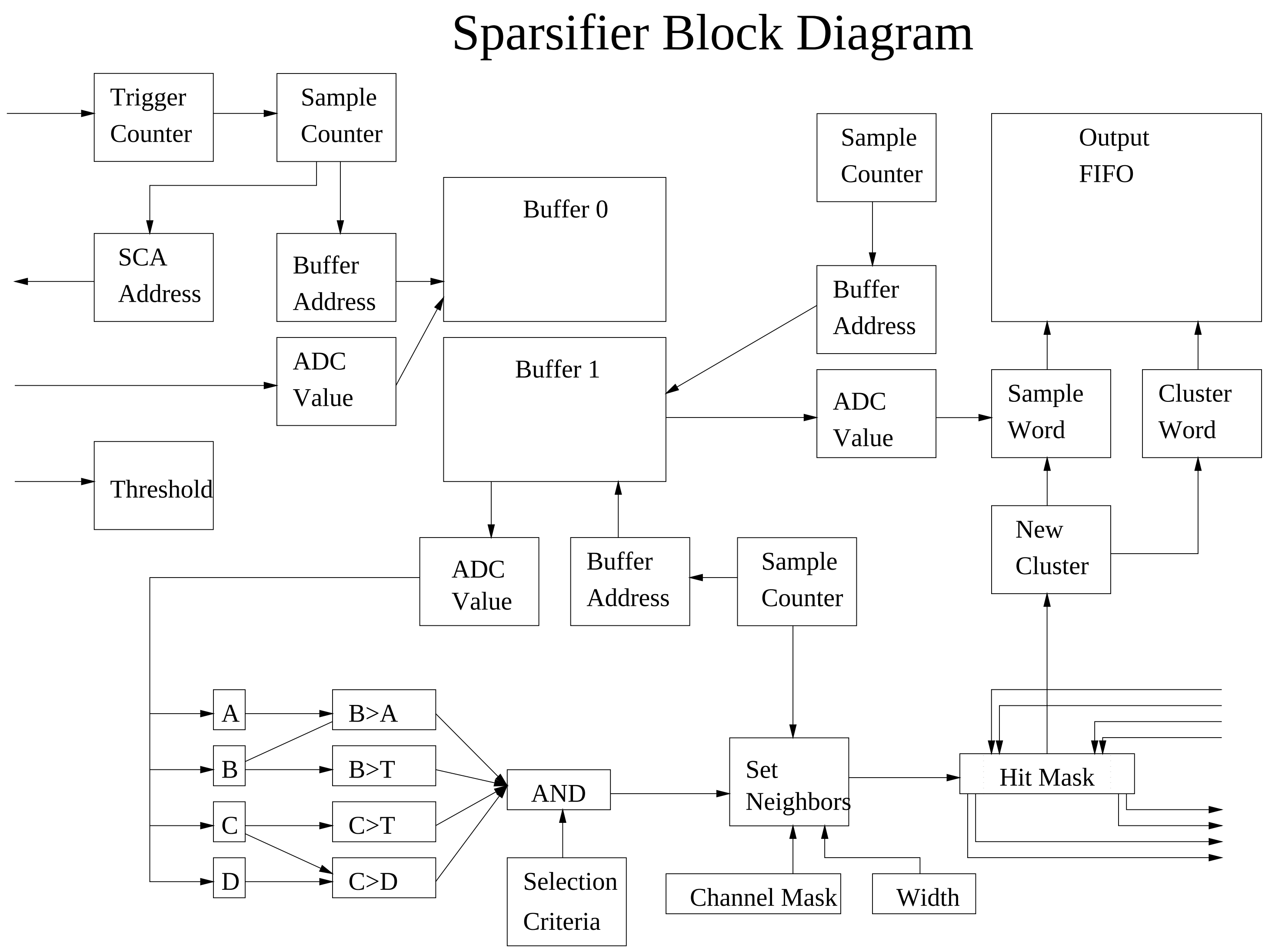

Figure 1: Block diagram of the sparsification implementation.